Bachelor's Thesis (TUAS)

Degree Program: Information Technology

Specialization: Internet Technology

2013

Joseph Muli

ANDROID AS A PLATFORM FOR DATABASE APPLICATION DEVELOPMENT

– CASE: WINHA MOBILE

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Internet Technology

14.2.2013 | 42 pages

Instructor: Tiina Ferm

Joseph Muli

ANDROID AS A PLATFORM FOR DATABASE APPLICATION DEVELOPMENT:

CASE: WINHA MOBILE

This thesis aims to help beginner Android developers and anyone interested in database intergration on Android Operating System understand the basic fundamentals on how to design database applications, specifically for mobile devices.

A review of Android applications has been made to give an overview of the general properties of the applications in relation to database creation and management. To accomplish this thesis, SQL (Structured Query Language) and Android application development were studied.

As a result of this thesis, a case study Android application (Winha Mobile) has been developed to compliment the review. Winha Mobile was designed and developed in Eclipse environment using Java as the programming language and SQLite as the database. After testing Winha Mobile, courses and their subsequent grades were inserted successfully and could be viewed in the database. A conclusion made in this thesis is that, database application development in Android operating system can be perfected for commercial purposes as well as for learning.

KEYWORDS:

(Android, Eclipse and SQLite)

FOREWORD

I would like to thank God for making it possible for me to handle this report smoothly. I would also like to thank Mrs. Tiina Ferm for having the time to supervise and guide me. Many thanks to Bucky-Roberts for his in depth android tutorials which helped me a great deal. Last but not least, my family for encouraging and believing in me through the start to finish.

Spring 2013

Joseph Kyalo Muli

TABLE OF CONTENTS

6 TABLES

7 FIGURES

REFERENCES

ACRONYMS, ABBREVIATIONS AND SYMBOLS

SQL	Structured Query Language
AVD	Android Vertual Device
SDK	Software Development Kit
API	Application Programming Interface
GUI	Graphical User Interface

1 Introduction

Databases are an essential part at an era where substantial information can be stored in a single file or a device. Android, an operating system by Google Inc. has arguably the largest market share on smartphones and tablets with the highest number of units sold worldwide [1].

Android is the most popular mobile platform in the world with a current 1.5 billion downloads a month and growing [1]. Open distribution of applications on the platform has led to this growth.

With smartphones being able to process many applications simultaneously, a majority or rather all mobile applications have an underlying database to store the required information. Consumers are not limited to any mobile device in the market hence most applications can be developed on different frameworks and be able to be ported on an Android device.

Android uses SQLite to manage its database. SQLite is based on SQL (Structured Query Language) syntax to manage data in devices like mobile phones and tablets. As the name suggests, it is very light [2]. SQLite helps in the creation of a database, insertion of records, updating of records and deleting of records.

More complex database applications are being currently deployed on Android Operating System, thus making it interesting for developers. Google Inc, the company behind Android has provided quite a variety of SDK (Software Developement Kit) for different computer operating systems including Windows, Linux and Macintosh. Resources are also available [3] for developers to get to know of how to develop applications on it as well as earning a living by selling them in the Android Market, now called Google Play. Since the concept is the same, various user interfaces and functionalities can be implemented.

This thesis is divided into several chapters, each discussing the general process of the project. The chapters are summarized as follows:

Chapter 1 - Introduction

This chapter gives a summary of the whole thesis. It explains in detail the aim of this undertaking and the goals behind it.

Chapter 2 - Android Applications

In this chapter, the android operating system will be introduced to the reader so that the reader gets an idea of what it is all about. Special emphasis is given on SQLite which will make the reader understand the theory behing database management in Android operating system.

Chapter 3 - Design and implementation

The reader will get a chance to understand how to install the software needed to develop applications on Android operating system. In addition, he or she will learn how to deign the graphical user interface of the project with a matching underlying database design based on Java language using SQL syntax.

Chapter 4 - Running the application

Having the application ready, this chapter explains the different ways to test and run the application.

Chapter 5 - Conclusion

This chapter discusses if the aims of the project were met, what improvements need to be made as well as how to make such applications commercially available.

2 Android Applications

An Android application is a program which is designed to run on any device using the Android operating system.

2.1 Android Operating System

The Android operating system is a Linux based system mainly used for running mobile devices using touchscreen capability, for example, smartphones and tablets [4].

2.2 Application Fundamentals

When developing an application, the sourcecode is compiled with the Android SDK tools to a single package with an .apk suffix. During the packaging, data and resource files are also included. The .apk file acts as an installation program on Android-powered devices.

On installation, an application will be confined securely in a sandbox. The Android operating system is a multi-user Linux system which assigns a unique ID (known to the application only). The system sets permissions for all the files so that the user ID assigned to a particular application is the only one able to access them. An application code runs in isolation from other applications because by default every application runs its own Linux process.

Access to Android applications is given to the components which have specific authentication. This ensures a secure environment whereby an application cannot access parts of the system for which it is not given permission (principle of least privilege). When applications share data with others to access system services, the applications must bare the same certificate. Permissions to access device data should be granted by the user at the time of installation.

2.3 Application Components

Application components are the essential bulding blocks of an Android application. There are four types of application components: [5]

1. Activities are components which interact to the user though the screen. For example, contacts can have activities like view contact, edit contact and delete contact.

2. Services are components that run in the background without user interfaces. For example, music may play in the background while another application is on use through the screen.

3. Content providers are components that enable applications to share data. For example contacts can be made available to other applications.

4. Broadcast receivers are components that respond to external events. They might do these by creating status bar notifications to make alerts, for example, when a phone is ringing.

Components are activated through intents. Activity intents never overlap with services nor broadcast receivers. An intent is a description of an operation to be performed. (Figure 1.0)

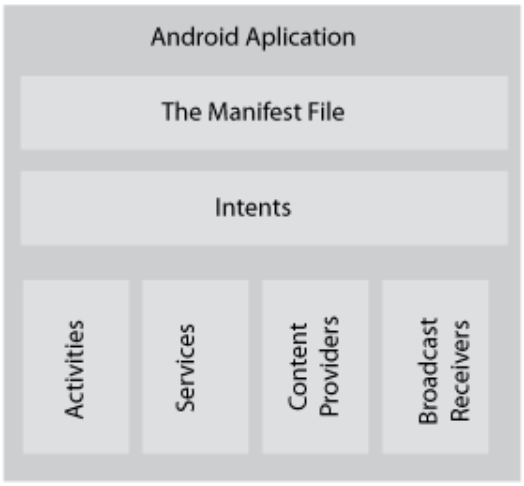

Figure 1. Android application components

2.4 The Manifest File

The Android Manifest file always bears the name AndroidManifest.xml and is located at the root of the application project directory. Before an application can start, the system

must know that the component exists by reading the manifest file. By doing this, the manifest file primarily informs the system about the application's components. (Appendix 1.4)

In addition to declaring the application's components, the Manifest file:

1. Identifies any user permissions the application requires, such as Internet access or read-access to the user's contacts.
2. Declares the minimum API Level required by the application, based on which APIs the application uses.
3. Declares hardware and software features used or required by the application, such as a camera, bluetooth services, or a multitouch screen.
4. Links API libraries of the application together (other than the Android framework APIs), such as the Google Maps library.

2.5 Application Resources

Resources like images, arrays, colors, string values, dimensions, animations, styles, menu, layout and others are found in a subdirectory of the project called res/. whereby they are externalized to maintain their independence. The subdirectories group resources by type and configuration.

Resouces can be accesed using resource IDs that are generated in the project's R class. The R class is a class contained in the Android API(Application Programming Interface) which holds information about elements.

Example.

android:id="@+id/tableLayoutExample"

The above layout will be accessed using the ID called tableLayoutExample. On compilation of an application, R class is generated, which contains resource IDs for all the resources in your res/ directory.

Device configurations can change for example when the screen orientation moves from landscape to portrait or when the keyboard has to be hidden. At this point Android

restarts the running activity so that it reloads the application with alternative resources to match new device configuration. The previous state of the activity on restart is restored through the normal Activity lifecycle.

2.6 User Interface

The Android user interface is made up of many componets. They are as follows:

1. Layout - The application's visual structure can be viewed in several layout views namely linear layout, relative layout, list view and grid view. (Figure 2.0)

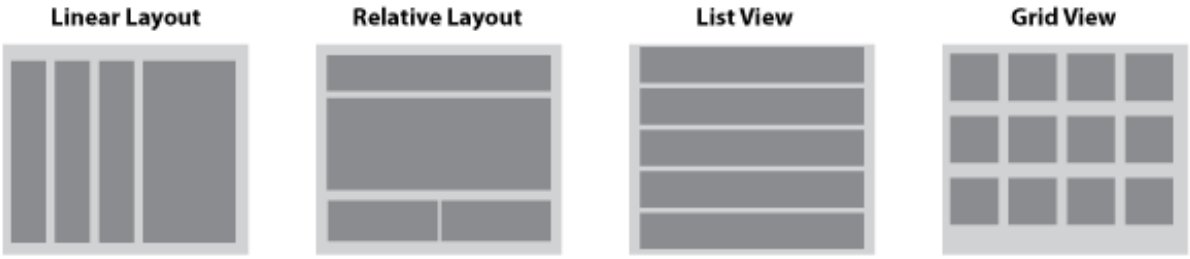

Figure 2. Layout views

2. User interaction to the device - Users have a variety of controls and input methods. Controls include buttons, spinners, toggle buttons, radio buttons and checkboxes. Input methods are mainly text fields. Menus will also lie under this category. Activities of an application can dictate what certain menus will do.

3. Information/warning to the user - In case there is an error or a notification has to be displayed by the user, dialogs are used, notifications(mostly in the notification bar) and toasts(dark boxed alerts).

4. Search - By default, Search is the core user feature of Android. The user has a text field to type and a button to execute the search of the data needed. Android has also incorporated voice search capability based on internet connectivity.

5. Settings - A user has the freedom to make a choice of how a certain application needs to respond to certain changes.

2.7 Android Database Package

Android database packages are contained in the API (Application Programming Interface) based on the API levels which are dictated by the release of any upgrade on the software. The API level used for this thesis report is Level 16. Android.database, android.database.sqlite and javax.sql are the main packages which handle databases.

1. android.database - It is mostly used to manage data in public databases.
2. android.database.sqlite - It is used to manage data in private databases.
3. javax.sql - It uses Java Language to interact with the database environment.

Each class in use manages the database with the aid of a cursor object which makes it possible to interact with the user interface. The classes manage database creation, open and close the databases, compile SQLite programs as well as execute SQL queries to perform specific operations on the database.

The Database helper class is defined by extension from the SQL OpenHelper class. This helps in creating or opening a database, upgrading the database, defining tables, columns, indexes, triggers and views of a database.

A database adapter class is designed where the basic CRUD (Create, Read, Upgrade and Delete) database management tools are programmed. The SQl queries are used to manage CRUD operations [6]. The data retrieval is done by the use of cursors. The methods used in the Helper class are the following:-

1. Insert() - This method is used to insert or put rows into a database.
2. query() - This method makes requests for matching or similar data in the database.
3. update() - This method is used to make replacements to rows based on the specified request.
4. delete() - This method is used to remove rows or data from the database.

2.8 SQLite

SQLite is the smallest SQL database engine in the world. It has the following properties [2].

1. Serverless – No process is running for the database. It is driven by a library.
2. Self-contained – It requires very little support from the operating system.
3. Zero-configuration – No configuration is required.
4. Transactional – All changes and queries are Atomic, Consistent, Isolated and Durable(ACID).

Android contains the SQLite3 version of SQLite to manage and create databases.

2.9 Data Storage

When dealing with databases, the data needs to be saved so that it can be accessed when neeeded.

There are several options to save data in Android, depending on specific user needs such as availability of the data to other applications or making it private and the space the data will occupy. The following are data storage options [7]:

- External storage - It stores public data on shared external storage e.g sd cards.
- Internal Storage - It stores public data on the device memory.
- Network Connection - It stores data on the web with the user's network server.
- Shared preferences - This option stores private primitive data in key-value pairs e.g strings and arrays.
- SQLite Databases - This option stores structured data in a private database.

3 Design and Implementation

Winha Mobile is a mock student grading application. A course name is inserted by the user with a subsequent grade. It is a simple database application which uses basic SQL (Structured Query Language) operations. This application is a product of the thesis showcasing in detail how databases are created and maintained in the Android Operating System.

The project was developed on both Macintosh and Windows operating system with the following software used compartible to the respective operating system. (Table 1.0)

Software	**Description**
Eclipse	A java software development environment
Android SDK	Android software development kit

Table 1.0 Software used

Though Eclipse is the officially supported environment to develop android applications, NetBeans IDE (Integrated Development Environment) also supports android application development through a plugin.

3.1 Installation of Eclipse and Android SDK

The following steps will be needed to install Eclipse and Android SDK. The version of Eclipse will depend on the requirements of the user. The download files will vary from one perating system to another. Download:

1. Eclipse from: www.eclipse.org/downloads
2. Android SDK from http://developer.android.com/sdk/installing/index.html
3. Install the Eclipse ADT(Android Development Tools) plugin by following the steps in the following link.

   http://developer.android.com/sdk/installing/installing-adt.html

3.2 Programming Language

By default, the Android programming language is Java. It has the following advantages.

1. Platform-independent - Java is able to be transferred easily from one computer system to another.
2. Easy to learn - It is simple to learn, write, debug and compile compared to other languages.
3. Object-oriented - It is centered around objects and how to make the work together.
4. Security - Java was designed with security in mind making it secure for running.
5. Multithread - It is able to perform several tasks simultaneously within a program.

3.3 GUI (Graphical User Interface) Design

The home screen of Android phone has the status bar which shows the time, network carrier name, battery level, the search box and some application icons. (Figure 3)

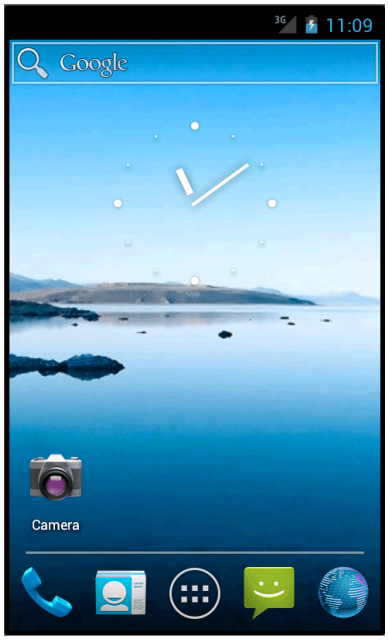

Figure 3. Home screen of an Android phone (Running on Ice Cream Sandwich Version 4.0)

The graphical user interface is a vital aspect of any application. Through this, users are able to have a rough idea of what the application can do.

Launcher icons

All Android applications have launcher icons. Launcher icons are clicked so that the application can run. (Figure 4) Launcher icons have properties which include:

- ic_launcher prefix.
- .png extension.
- 36, 48, 72, 96 pixels wide and can be scaled in that ratio.
- Located in the /res folder of an application.

The design of the icons is performed using image-editing software such as Adobe Photoshop. Vector shape use is highly recommended so that upon scaling, no detail is lost.

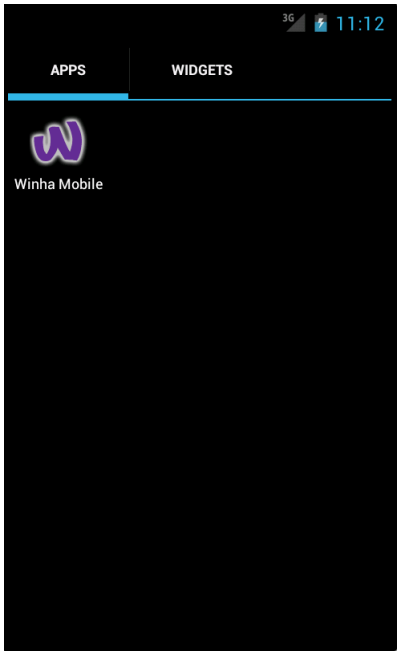

Figure 4. Laucher icon

The thesis project (Winha Mobile) is based on a simple approach to insert, update, view and delete data in a database. The user can insert a course and grade it and perform any alterations thereafter. (Figure 5)

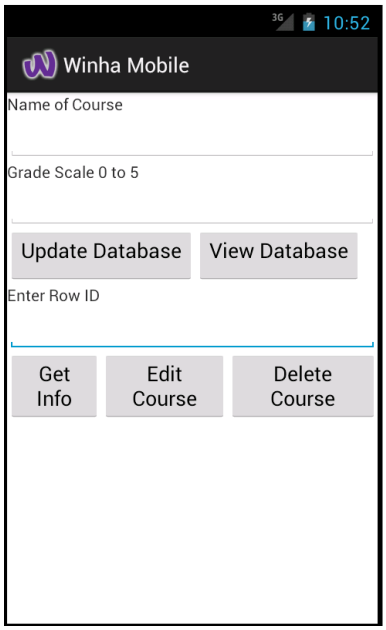

Figure 5. Winha Mobile Application

The views which open upon running an application are designed in two options. They can be purely written in xml code and viewed using the graphical viewer or elements can be dragged and dropped on the design window and placed in desired positions. Xml views are located or created in layout folder. (Figure 6)

Figure 6. xml design options

The code below corresponds to the activity_main.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world" />

</RelativeLayout>
```

3.4 Database Design

The course database has a table with the Id, course name and course grade. The Id is the primary key and of integer type with an auto-increment function. The course name and grade are of text type. (Figure 7).

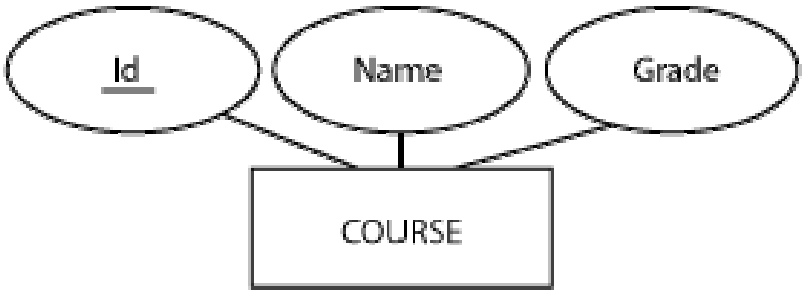

Figure 7. Entity Relational Diagram

Creating the database with SQLite (Appendix 1.1 Grade.java).

```
private static final String DATABASE_NAME ="CourseGradedb";
private static final String DATABASE_TABLE ="courseTable";
private static final int DATABASE_VERSION =1;
```

The database for the course will have a name and a table with three fields namely: Id ,Name and Grade.

```
public static final String KEY_ROWID ="_id";
public static final String KEY_NAME ="course_name";
public static final String KEY_GRADE ="course_grade";
```

By default, SQLite uses an invisible ROWID to identify each row in a table uniquely.

A method called exec.SQl is used to manipulate SQL queries.

```
db.execSQL("CREATE TABLE " + DATABASE_TABLE + " (" +
KEY_ROWID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
KEY_NAME + " TEXT NOT NULL, " +
KEY_GRADE + " TEXT  NOT NULL);"
```

```
);
```

DBelper method is used to open the database.

```
public Grade open() throws SQLException{
                        myHelper = new DbHelper(myContext);
                        myDatabase = myHelper.getWritableDatabase();
                        return this;
                        }

public long createEntry(String name, String grade) {
                        // TODO Auto-generated method stub
                        ContentValues cv = new ContentValues();
                        cv.put(KEY_NAME, name);
                        cv.put(KEY_GRADE, grade);
                        return myDatabase.insert(DATABASE_TABLE, null, cv);
            }
```

The data is accessed using a Cursor factory.

```
public String getData() {
                        // TODO Auto-generated method stub
                        String[] columns = new String[]{ KEY_ROWID, KEY_NAME,
KEY_GRADE};
                        Cursor c = myDatabase.query(DATABASE_TABLE, columns,
null, null, null, null, null);
                        String result = "";

                        int iRow = c.getColumnIndex(KEY_ROWID);
                        int iName = c.getColumnIndex(KEY_NAME);
                        int iGrade = c.getColumnIndex(KEY_GRADE);

                        for (c.moveToFirst(); !c.isAfterLast();
c.moveToNext()){
                                    result = result + c.getString(iRow) + " "
+ c.getString(iName) + " " + "\t\t\t\t" + c.getString(iGrade) + "\n";
                                    }
                        return result;
            }
```

The interaction between the interface and the database is made possible by linking the layouts and the components (Appendix 1.2 SQLiteExample.java).

```
protected void onCreate(Bundle savedInstanceState){
            //TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            setContentView(R.layout.sqliteexample);
            sqlUpdate = (Button) findViewById(R.id.bSQLUpdate);
            sqlName = (EditText) findViewById(R.id.etSQLName);
            sqlGrade = (EditText) findViewById(R.id.etSQLGrade);

            sqlView = (Button) findViewById(R.id.bSQLopenView);
            sqlView.setOnClickListener(this);
```

```
            sqlUpdate.setOnClickListener(this);

            sqlRow = (EditText)findViewById(R.id.etSQLrowInfo);
            sqlModify = (Button)findViewById(R.id.bSQLmodify);
            sqlGetInfo = (Button)findViewById(R.id.bgetInfo);
            sqlDelete = (Button)findViewById(R.id.bSQLdelete);
            sqlDelete.setOnClickListener(this);
            sqlModify.setOnClickListener(this);
            sqlGetInfo.setOnClickListener(this);
}
```

The source code for the project is in Appendix 1.0 Winha Mobile.

4 Testing the Application

After writing the source code for an application [8] and fixing all bugs, an application is ready to be tested.

4.1 Android Virtual Device(Emulator)

An emulator is a virtual look-alike of an Android phone used to test run applications on Eclipse.

After installing Eclipse and the Android SDK, an emulator can be created based on specifications needed by the application being designed, e.g., touch capability. The AVD (Android Virtual Device) is created by accessing windows > AVD manager in Eclipse (Figure 8 and Figure 9).

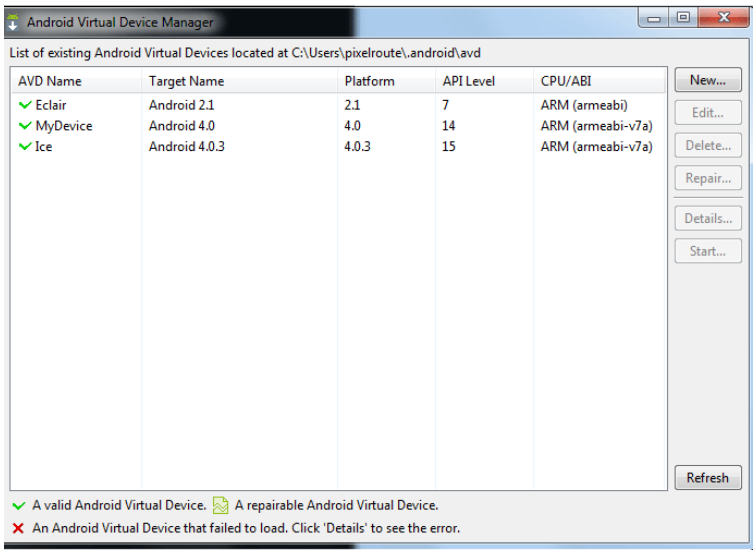

Figure 8. Creating a new AVD

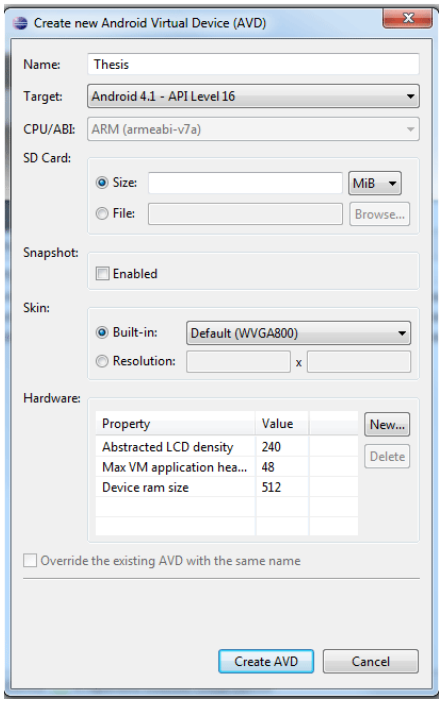

Figure 9. Naming and choosing AVD specifications

Once the AVD(Android Virtual Device) has been created, we click start to open it. (Figure 10)

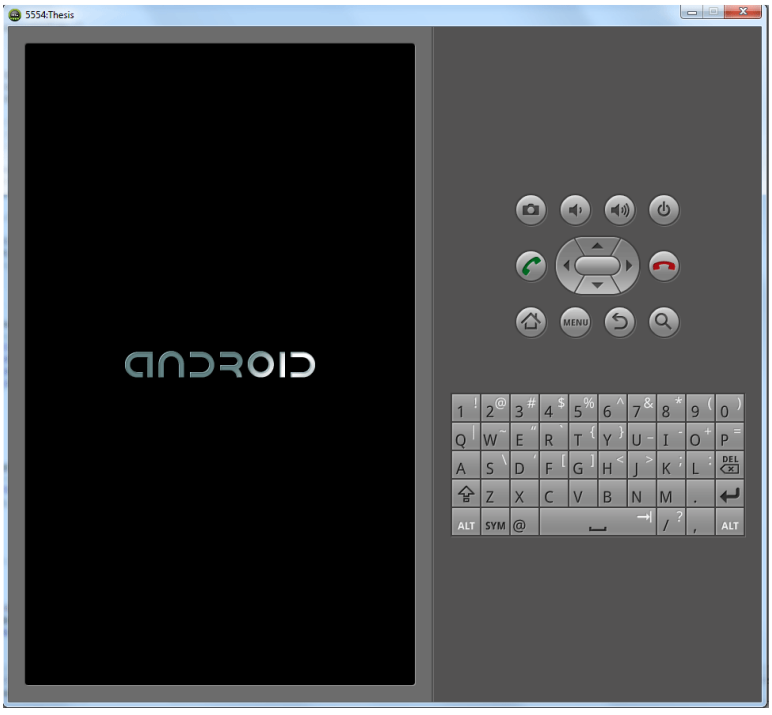

Figure 10. Starting the AVD

When view database button is clicked (Figure 5), the contents in the database are displayed (Figure 11).

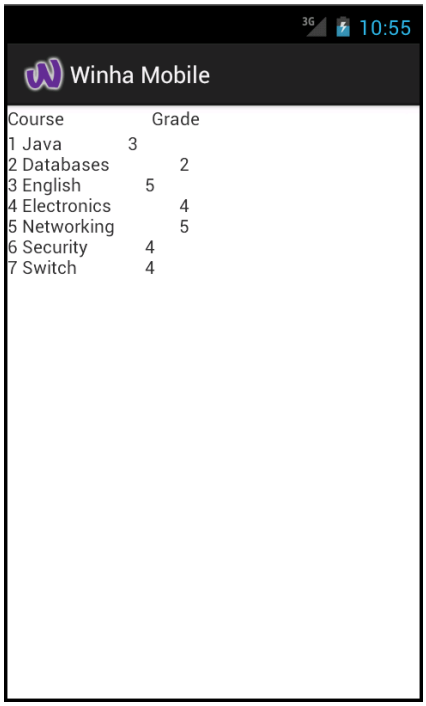

Figure 11. Results of clicking view database button

Data can be inserted in the database by placing the cursor to the ‘Name of course’ field and typing the name of the course and typing the grade to the ‘Grade Scale 0 to 5’ field (Figure 12).

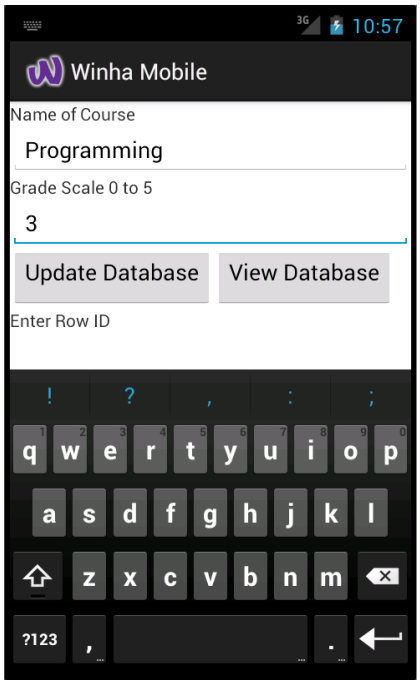

Figure 12. Inserting a new course and grading it

Once a new course and grade have been inserted, on clicking update database, a dialog will pop up to show that either the insert was successful or not (Figure 13).

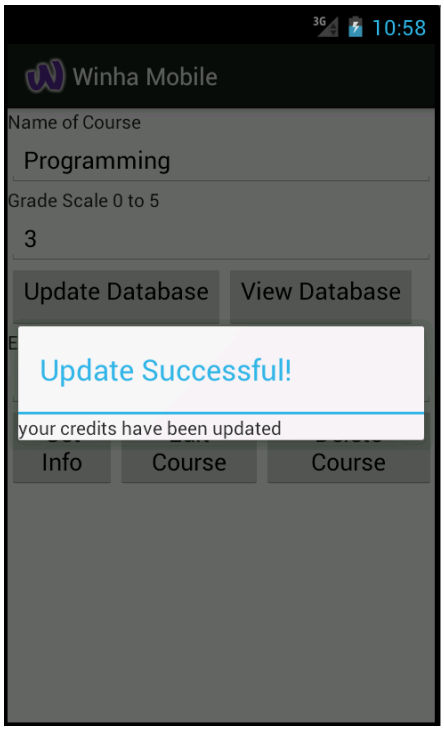

Figure 13. A Dialog display of successful insertion

When a user wants to retrieve a specific record from the database, they have to put the Id which is unique (Figure 14 and Figure 15). A list of the data can be accessed by pressing ‘View Database’ button.

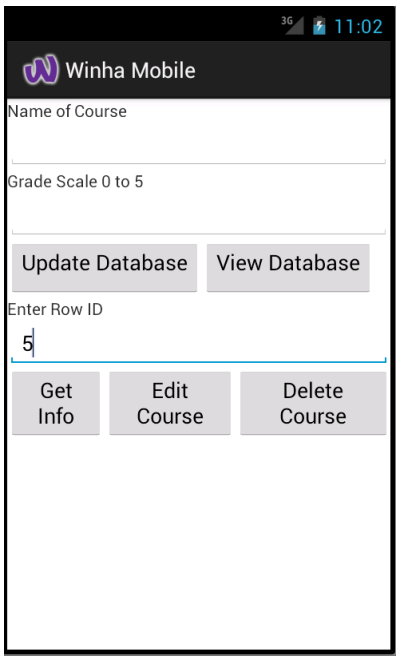

Figure 14. Fetching information using the Primary key (Row_Id) by pressing Get Info button

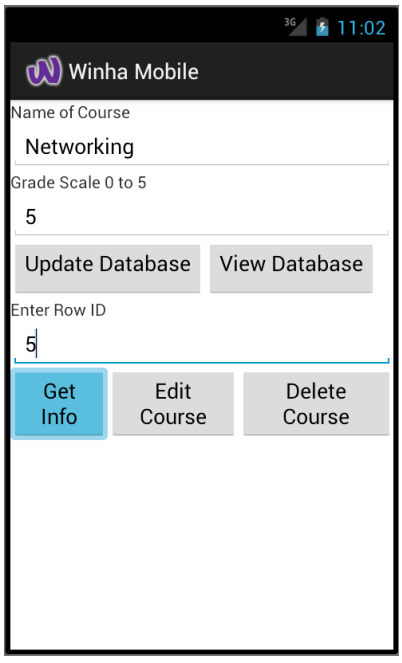

Figure 15. Results after pressing Get Info button

When deleting a record, first the user has to enter the row and fetch the data from the database by pressing ‘Get info’ button then after by pressing ‘Delete Course’ button (Figure 16).

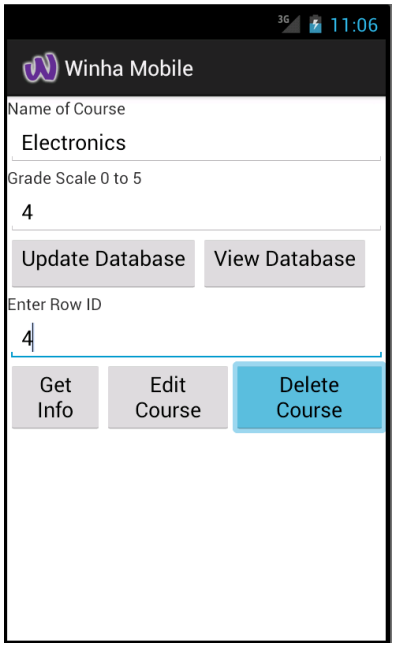

Figure 16. Deleting a course

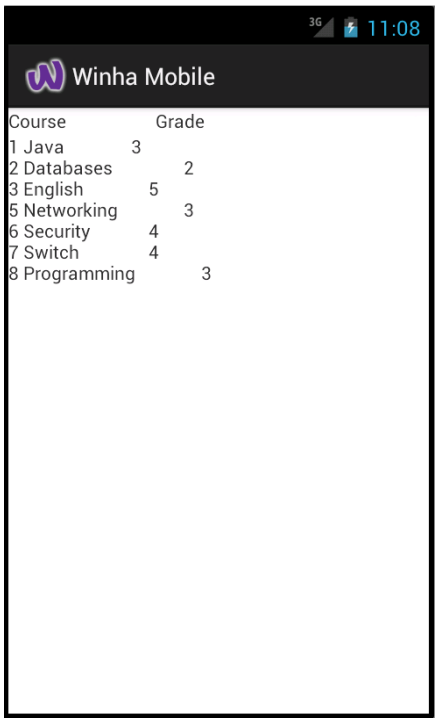

Figure 17. Course deleted (Row_Id 4)

When an incomplete operation or a wrong operation is made, a dialog pops up with an error message (Figure 18).

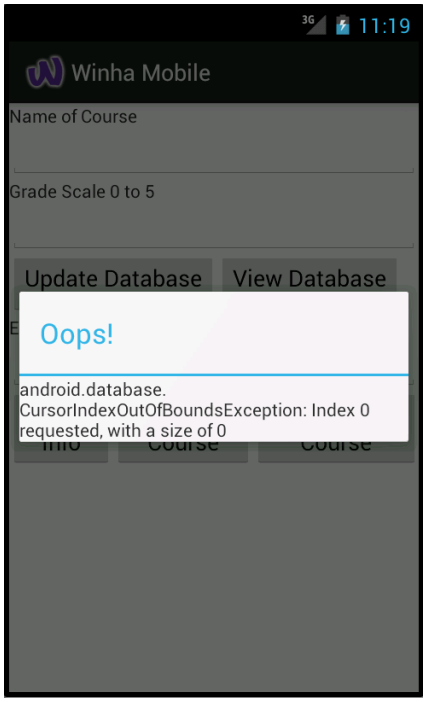

Figure 18. A dialog display for error exceptions

4.2 Mobile Device

Debbuging can be done on most Android devices. A guide on how to set preferences and can be found in www.developer.android.com/tool/device.html. When developing Android applications, the project contains a .apk file which can be ported in a mobile device running on android. The file is unpackaged by the phone as the application gets installed (Figure 19).

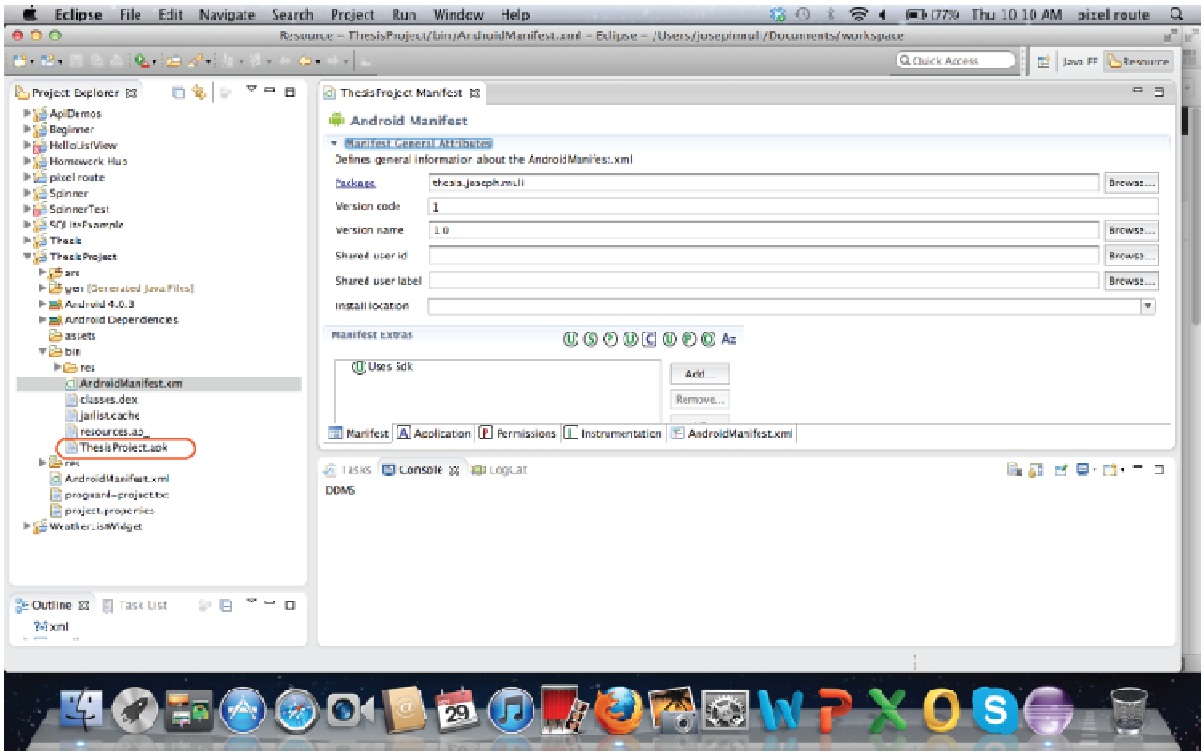

Figure 19 The .apk file

5 Conclusion

Results

The project was tested on an emulator as well as an Android mobile device (HTC Sensation). Records could be inserted, updated, viewed and also deleted. The goal was met, whereby a database was created and some basic SQL (Structured Query Language) operations were executed. A course name and a corresponding grade were inserted successfully into the database. The application ran smoothly and the interface components responded as expected. The size of the text and buttons changed slightly upon installing to the mobile device. Some challenges arose with the ROW_ID. It was realized that, by default android has ROW_ID for unique access to the table.Naming for the ROW_ID was omitted in the project by choice.

Reccomendations

The application is designed for basic purposes though it can be developed for more complex tasks. The idea of Winha Mobile primarily is to be used in a school or institution environment. Improvements such as login access can be created for the users so that they can view their courses and grades as well as do other operations such as enrolling for courses, registering for resits and also accessing the email and curriculum resource portal. The login access will enforce security to the application since the users will be granted any necessary priviledges to the database.

There are ways of defining where the database will be saved on creation. This should be specified since some databases hold a lot of data hence they might need a larger storage location as opposed to a default phone memory or cache.

Database design based on Android platform has a commercial worth for both developers and institutions designing and implementing such applications. Google Play is a service available to developers where they can register their applications and be able to make money though purchases made by users of Android Operating System. Currently, Google Play has surpassed a 600,000 applications and games in its store with more than 20 million users of Android since it began.

References

[1] www.developer.android.com/about/index.html. Accessed: November 24th 2012

[2] www.sqlite.org. Accessed: November 2nd 2012

[3] www.developer.android.com. Accessed: November 2nd 2012

[4] http://en.wikipedia.org/wiki/Android_(operating_system). Accessed: November 24th 2012

[5] www.developer.android.com/guide/components/fundamentals.html. Accessed: November 24th 2012

[6] Using SQlite, O'Reilly by Jay A. Kreibich 2010

[7] www.developer.android.com/guide/topics/data-storage..html. Accessed November 24th 2012

[8] www.thenewboston.org. Accessed: October 16th 2012

Appendix 1.0 **Winha Mobile**

Appendix 1.1 **Grade.java**

```
package com.example.yuyi;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class Grade {

public static final String KEY_ROWID ="_id";
public static final String KEY_NAME ="course_name";
public static final String KEY_GRADE ="course_grade";

private static final String DATABASE_NAME ="CourseGradedb";
private static final String DATABASE_TABLE ="courseTable";
private static final int DATABASE_VERSION =1;

private DbHelper myHelper;
private final Context myContext;
private SQLiteDatabase myDatabase;

private static class DbHelper extends SQLiteOpenHelper{

public DbHelper(Context context) {
super(context, DATABASE_NAME, null, DATABASE_VERSION);
```

```
// TODO Auto-generated constructor stub

}

@Override
public void onCreate(SQLiteDatabase db) {
// TODO Auto-generated method stub
db.execSQL("CREATE TABLE " + DATABASE_TABLE + " (" +
KEY_ROWID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
KEY_NAME + " TEXT NOT NULL, " +
KEY_GRADE + " TEXT  NOT NULL);"
);



}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
// TODO Auto-generated method stub
db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);
onCreate(db);

}
}

public Grade(Context c){
myContext = c;
}

public Grade open() throws SQLException{
myHelper = new DbHelper(myContext);
```

```
myDatabase = myHelper.getWritableDatabase();

return this;

}


public void close(){

myHelper.close();

}


public long createEntry(String name, String grade) {

// TODO Auto-generated method stub

ContentValues cv = new ContentValues();

cv.put(KEY_NAME, name);

cv.put(KEY_GRADE, grade);

return myDatabase.insert(DATABASE_TABLE, null, cv);

}


public String getData() {

// TODO Auto-generated method stub

String[] columns = new String[]{ KEY_ROWID, KEY_NAME, KEY_GRADE};

Cursor c = myDatabase.query(DATABASE_TABLE, columns, null, null, null, null,
null);

String result = "";


int iRow = c.getColumnIndex(KEY_ROWID);

int iName = c.getColumnIndex(KEY_NAME);

int iGrade = c.getColumnIndex(KEY_GRADE);


for (c.moveToFirst(); !c.isAfterLast(); c.moveToNext()){

result = result + c.getString(iRow) + " " + c.getString(iName) + " " +
"\t\t\t\t" + c.getString(iGrade) + "\n";

}

return result;
```

```
}

public String getName(long l) throws SQLException{
// TODO Auto-generated method stub
String[] columns = new String[]{ KEY_ROWID, KEY_NAME, KEY_GRADE};
Cursor c = myDatabase.query(DATABASE_TABLE, columns, KEY_ROWID + "=" + l,
null, null, null, null);
if (c != null){
c.moveToFirst();
String name = c.getString(1);
return name;
}
return null;
}

public String getGrade(long l) throws SQLException{
// TODO Auto-generated method stub
String[] columns = new String[]{ KEY_ROWID, KEY_NAME, KEY_GRADE};
Cursor c = myDatabase.query(DATABASE_TABLE, columns, KEY_ROWID + "=" + l,
null, null, null, null);
if (c != null){
c.moveToFirst();
String grade = c.getString(2);
return grade;
}
return null;
}

public void updateEntry(long lRow, String mName, String mGrade) throws
SQLException{
// TODO Auto-generated method stub
ContentValues cvUpdate = new ContentValues();
```

```
cvUpdate.put(KEY_NAME, mName);
cvUpdate.put(KEY_GRADE, mGrade);
myDatabase.update(DATABASE_TABLE, cvUpdate, KEY_ROWID + "=" + lRow, null);

}

public void deleteEntry(long lRow1) throws SQLException{
// TODO Auto-generated method stub
myDatabase.delete(DATABASE_TABLE, KEY_ROWID + "=" + lRow1, null);

}

}
```

Appendix 1.2 SQLiteExample.java

```
package com.example.yuyi;

import android.app.Activity;
import android.app.Dialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class SQLiteExample extends Activity implements OnClickListener {

Button sqlUpdate, sqlView, sqlModify, sqlGetInfo, sqlDelete;
EditText sqlName, sqlGrade, sqlRow;

@Override
protected void onCreate(Bundle savedInstanceState){
//TODO Auto-generated method stub
super.onCreate(savedInstanceState);
setContentView(R.layout.sqliteexample);
sqlUpdate = (Button) findViewById(R.id.bSQLUpdate);
sqlName = (EditText) findViewById(R.id.etSQLName);
sqlGrade = (EditText) findViewById(R.id.etSQLGrade);

sqlView = (Button) findViewById(R.id.bSQLopenView);
sqlView.setOnClickListener(this);
```

```
sqlUpdate.setOnClickListener(this);

sqlRow = (EditText)findViewById(R.id.etSQLrowInfo);
sqlModify = (Button)findViewById(R.id.bSQLmodify);
sqlGetInfo = (Button)findViewById(R.id.bgetInfo);
sqlDelete = (Button)findViewById(R.id.bSQLdelete);
sqlDelete.setOnClickListener(this);
sqlModify.setOnClickListener(this);
sqlGetInfo.setOnClickListener(this);
}

public void onClick(View arg0) {
//TODO Auto-generated method stub

switch (arg0.getId()){

case R.id.bSQLUpdate:

boolean didItWork = true;
try{
String name = sqlName.getText().toString();
String grade = sqlGrade.getText().toString();

Grade entry = new Grade(SQLiteExample.this);
entry.open();
entry.createEntry(name, grade);
entry.close();

}catch (Exception e){
didItWork = false;
String Error = e.toString();
```

```
Dialog d = new Dialog(this);
d.setTitle("Oops!");
TextView tv = new TextView(this);
tv.setText(Error);
d.setContentView(tv);
d.show();
}finally{
if (didItWork){
Dialog d = new Dialog(this);
d.setTitle("Update Successful!");
TextView tv = new TextView(this);
tv.setText("your credits have been updated");
d.setContentView(tv);
d.show();

}

}

break;
case R.id.bSQLopenView:
Intent i = new Intent("com.example.yuyi.SQLVIEW");
startActivity(i);
break;
case R.id.bgetInfo:
try{
String s = sqlRow.getText().toString();
long l = Long.parseLong(s);
Grade gr = new Grade(this);
gr.open();
String returnedName = gr.getName(l);
```

```
String returnedGrade = gr.getGrade(l);
gr.close();

sqlName.setText(returnedName);
sqlGrade.setText(returnedGrade);
}catch (Exception e){

String Error = e.toString();
Dialog d = new Dialog(this);
d.setTitle("Oops!");
TextView tv = new TextView(this);
tv.setText(Error);
d.setContentView(tv);
d.show();
}

break;

case R.id.bSQLmodify:
try{
String mName = sqlName.getText().toString();
String mGrade = sqlGrade.getText().toString();
String sRow = sqlRow.getText().toString();
long lRow = Long.parseLong(sRow);

Grade ex = new Grade(this);
ex.open();
ex.updateEntry(lRow, mName, mGrade);
ex.close();
}catch (Exception e){
```

```
String Error = e.toString();
Dialog d = new Dialog(this);
d.setTitle("Oops!");
TextView tv = new TextView(this);
tv.setText(Error);
d.setContentView(tv);
d.show();
}
break;

case R.id.bSQLdelete:
try{
String sRow1 = sqlRow.getText().toString();
long lRow1 = Long.parseLong(sRow1);
Grade ex1 = new Grade(this);
ex1.open();
ex1.deleteEntry(lRow1);
ex1.close();
}catch (Exception e){

String Error = e.toString();
Dialog d = new Dialog(this);
d.setTitle("Oops!");
TextView tv = new TextView(this);
tv.setText(Error);
d.setContentView(tv);
d.show();
}break;
}
}
}
```

Appendix 1.3 SQLView.java

```
package com.example.yuyi;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class SQLView extends Activity{

@Override
protected void onCreate(Bundle savedInstanceState){
//TODO Auto-generated method stub
super.onCreate(savedInstanceState);
setContentView(R.layout.sqlview);
TextView tv = (TextView) findViewById(R.id.tvSQLinfo);
Grade info = new Grade(this);
info.open();
String data = info.getData();
info.close();
tv.setText(data);

}

}
```

Appendix 1.4 AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.yuyi"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".SQLiteExample"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            </activity>
           <activity
            android:name=".SQLView"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="com.example.yuyi.SQLVIEW" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Appendix 1.5 sqliteexample.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

            <TextView android:id="@+id/textView1" android:text="Name of
Course"
        android:layout_width="wrap_content"
android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceSmall"></TextView>
    <EditText android:id="@+id/etSQLName" android:layout_width="fill_parent"
        android:layout_height="wrap_content"></EditText>
    <TextView android:text="Grade Scale 0 to 5" android:id="@+id/textView3"
        android:layout_width="wrap_content"
android:layout_height="wrap_content"></TextView>
    <EditText android:id="@+id/etSQLGrade" android:layout_width="fill_parent"
        android:layout_height="wrap_content"></EditText>
    <TableRow
    android:id="@+id/tableRow1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <Button android:text="Update Database" android:id="@+id/bSQLUpdate"
        android:layout_width="match_parent"
android:layout_height="wrap_content"
        android:layout_weight=".3"
android:gravity="center_horizontal"></Button>
    <Button android:text="View Database" android:id="@+id/bSQLopenView"
        android:layout_width="match_parent"
android:layout_height="wrap_content"
        android:layout_weight=".3"
android:gravity="center_horizontal"></Button>
    </TableRow>

    <TextView  android:text="Enter Row ID" android:id="@+id/textView2"
        android:layout_width="wrap_content"
android:layout_height="wrap_content"></TextView>
    <EditText  android:layout_width="match_parent"
        android:layout_height="wrap_content" android:id="@+id/etSQLrowInfo"
        android:inputType="number">
        <requestFocus></requestFocus>
    </EditText>
     <TableRow
    android:id="@+id/tableRow1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <Button android:text="Get Info" android:id="@+id/bgetInfo"
        android:layout_width="match_parent"
android:layout_height="wrap_content"
        android:layout_weight=".2"
android:gravity="center_horizontal"></Button>
    <Button android:text="Edit Course" android:id="@+id/bSQLmodify"
```

```
        android:layout_width="match_parent"
android:layout_height="wrap_content"
        android:layout_weight=".2"
android:gravity="center_horizontal"></Button>
    <Button android:text="Delete Course" android:id="@+id/bSQLdelete"
        android:layout_width="match_parent"
android:layout_height="wrap_content"
        android:layout_weight=".2"
android:gravity="center_horizontal"></Button>
    </TableRow>
 </LinearLayout>
```

Appendix 1.6 **sqlview.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="fill_parent"
   android:layout_height="wrap_content">

           <TableLayout
               android:id="@+id/tableLayout1"
               android:layout_width="fill_parent"
       android:layout_height="fill_parent" >
       <TableRow>
           <TextView
                       android:layout_width="fill_parent"
               android:layout_height="fill_parent"
                       android:text="Course"
                       android:layout_weight="0.2"
                       ></TextView>

           <TextView
                       android:layout_width="fill_parent"
                       android:layout_height="fill_parent"
                       android:text="Grade"
                       android:layout_weight="0.5"
                       ></TextView>

       </TableRow>
           </TableLayout>
          <TextView
                       android:id="@+id/tvSQLinfo"
                                   android:layout_width="fill_parent"
                       android:layout_height="fill_parent"
                       ></TextView>

</LinearLayout>
```